Generative classifiers

COMS 4771 Fall 2025

Classification problems

Problem: Create a program that, given an element from the input space $\mathcal{X}$, returns the element's corresponding label from the output space $\mathcal{Y}$

Classification problem: $\mathcal{Y}$ is discrete (and typically finite) set

Examples:

- Spam filtering

$$\mathcal{X} = \text{all possible emails}, \quad \mathcal{Y} = \{\text{ham}, \text{spam}\}$$

- Intrusion detection

$$\mathcal{X} = \text{all possible network traffic logs}, \quad \mathcal{Y} = \{\text{benign}, \text{malicious}\}$$

- News analysis

$$\mathcal{X} = \text{all possible news articles}, \quad \mathcal{Y} = \{\text{politics}, \text{sports}, \text{business}, \ldots\}$$

Model data as random variables

- Feature vector: random vector $X = (X_1, X_2, \ldots, X_d)$
- Label: a discrete random variable Y
- (X, Y) is called a random (labeled) example
- X and Y may be dependent!

Typical goal: create a classifier $f \colon \mathcal{X} \to \mathcal{Y}$ with low error rate

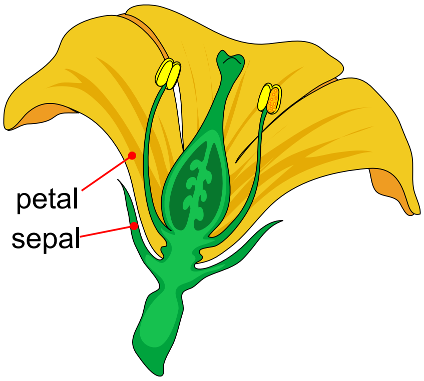
$$\mathrm{err}[f] = \Pr(f(X) \neq Y)$$

"Probability" is taken with respect to distribution of (X, Y)

Iris dataset

Iris dataset (Fisher, 1936)

- 3 classes of iris plants $\mathcal{Y} = \{\text{Setosa}, \text{Versicolor}, \text{Virginica}\}$
- Take some measurements of each iris plant



- Training data: 40 examples from each class; test data: 10 examples per class
- Problem: Create a program that, given the measurements of an iris plant, returns the class that the plant belongs to

Generative models for classification

Generative model (for classification): a family of probability distributions for (X, Y), each specified as follows.

- Specify marginal distribution p_Y of Y (class prior)
- For each $k \in \mathcal{Y}$, specify conditional distribution of X given $Y = k$ (class conditional distributions)

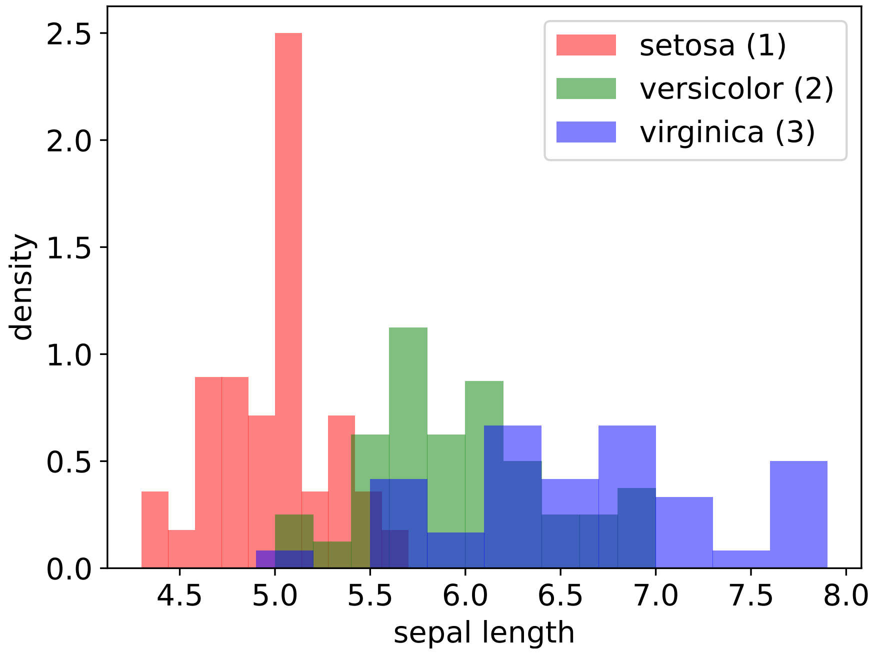
Example: Normal generative model

How to create classifier based on a distribution from the generative model?

- You have: $\hat{p}_Y$ and $\hat{p}_{X|Y=k}$ for each $k \in \mathcal{Y}$
- You want: $\hat{f} \colon \mathcal{X} \to \mathcal{Y}$

Generative model for iris dataset

Normal generative model for iris dataset using $x =$ sepal length



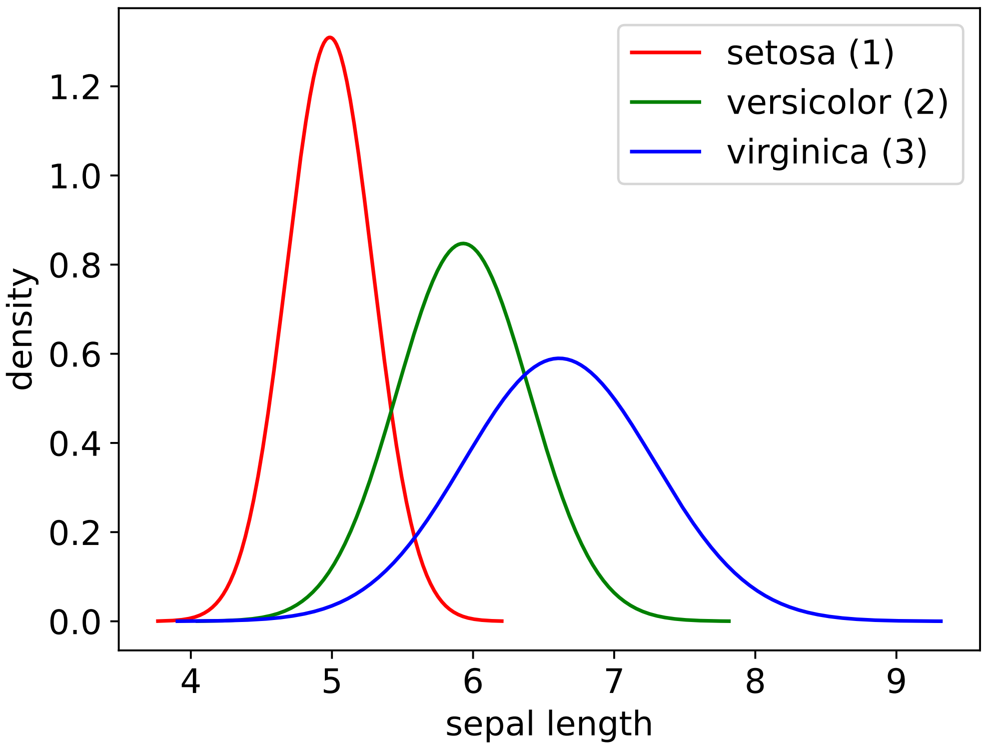
Maximum likelihood estimation (MLE) of π_k, μ_k, σ_k^2 for each $k \in \mathcal{Y}$:

$$\hat{\pi}_k = \frac{\#\text{ training examples with label } k}{\#\text{ training examples}}$$

$$\hat{\mu}_k = \text{average value of } x \text{ among examples with label } k$$

$$\hat{\sigma}_k^2 = \text{average value of } (x - \hat{\mu}_k)^2 \text{ among examples with label } k$$

k	setosa (1)	versicolor (2)	virginica (3)
$\hat{\pi}_k$	$1/3$	$1/3$	$1/3$
$\hat{\mu}_k$	4.99	5.93	6.61
$\hat{\sigma}_k$	0.31	0.47	0.68

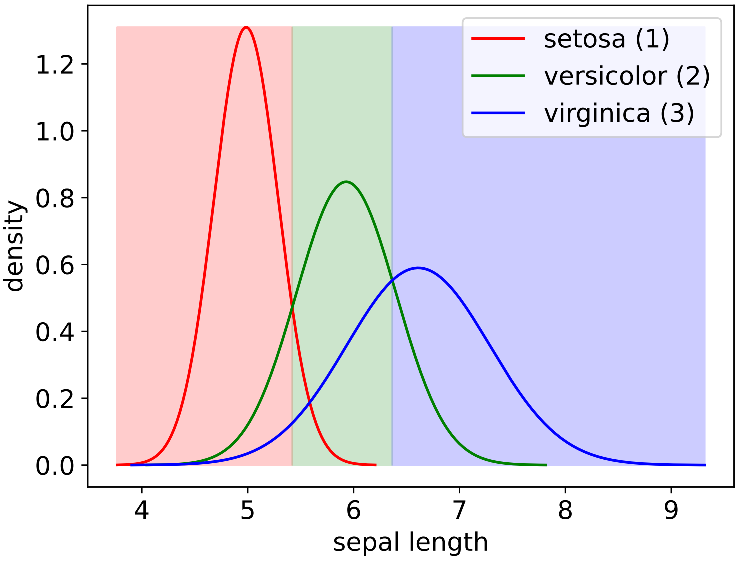


```python
def learn(train_x, train_y, num_classes=3):
  return [(np.mean(train_y == k), np.mean(train_x[train_y == k]),
   ↪ np.var(train_x[train_y == k])) for k in range(num_classes)]

def predict(params, test_x):
  log_posterior = np.array([np.log(prior) - np.log(sigma2) / 2 -
   ↪ (test_x - mu) ** 2 / (2*sigma2) for prior, mu, sigma2 in
   ↪ params])
  return np.argmax(log_posterior, axis=0)
```


Resulting classifier:

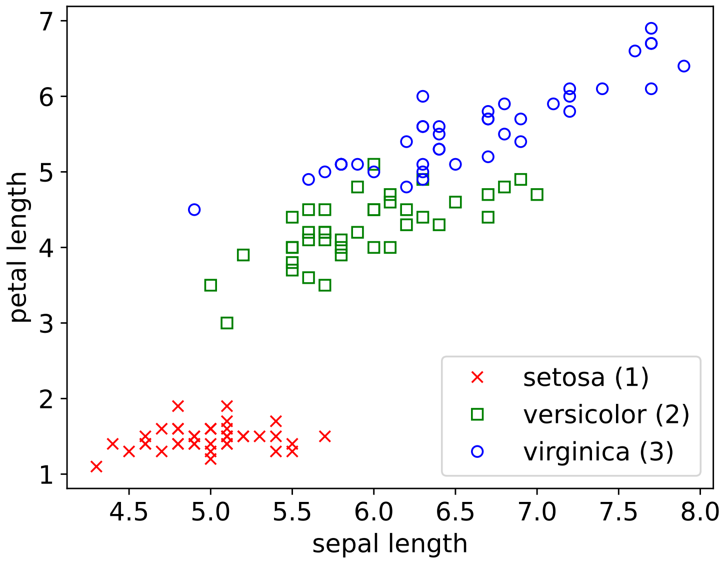
$$\hat{f}(x) = \arg\max_{k \in \mathcal{Y}} \hat{\pi}_k \cdot \frac{1}{\sqrt{2\pi\hat{\sigma}_k^2}} \exp\left(-\frac{(x-\hat{\mu}_k)^2}{2\hat{\sigma}_k^2}\right)$$



Training error rate of $\hat{f}$: 24%
Test error rate of $\hat{f}$: 40%

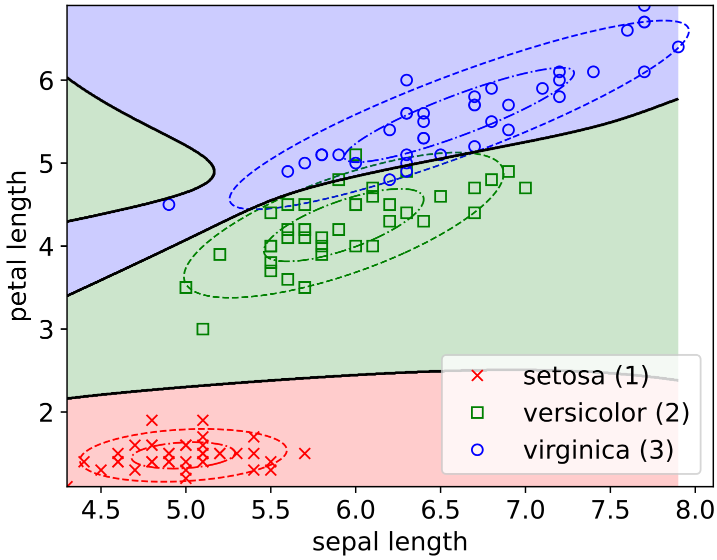
Bivariate normal distributions

Now use two features: $x = (x_1, x_2) = (\text{sepal length}, \text{petal length})$



Need generative model with class conditional distributions suitable for two-dimensional feature vectors

Bivariate normal distribution: 5 parameters (up from 2)



Resulting classifier has test error rate 10% (down from 40%)

Bivariate normal distribution with parameters $\mu_1, \mu_2, \Sigma_{1,1}, \Sigma_{1,2}, \Sigma_{2,2}$:

$$p_{(X_1,X_2)}(x_1, x_2) = \frac{1}{2\pi\sqrt{\det(\Sigma)}} \exp\left(-\frac{1}{2}\begin{bmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{bmatrix}^{\mathsf{T}} \Sigma^{-1} \begin{bmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \end{bmatrix}\right)$$

where

$$\Sigma = \begin{bmatrix} \Sigma_{1,1} & \Sigma_{1,2} \\ \Sigma_{2,1} & \Sigma_{2,2} \end{bmatrix}$$

is a positive definite matrix (and $\Sigma_{2,1} = \Sigma_{1,2}$).

Fitting bivariate normal distribution to data using MLE:

$$\hat{\mu}_1 = \text{average value of } x_1 \text{ in dataset}$$
$$\hat{\mu}_2 = \text{average value of } x_2 \text{ in dataset}$$
$$\hat{\Sigma}_{1,1} = \text{average value of } (x_1 - \hat{\mu}_1)^2 \text{ in dataset}$$
$$\hat{\Sigma}_{1,2} = \hat{\Sigma}_{2,1} = \text{average value of } (x_1 - \hat{\mu}_1)(x_2 - \hat{\mu}_2) \text{ in dataset}$$
$$\hat{\Sigma}_{2,2} = \text{average value of } (x_2 - \hat{\mu}_2)^2 \text{ in dataset}$$

(In context of generative models, do this for each class)

Multivariate normal distributions

General multivariate normal distribution in d-dimensions:

$$p_X(x) = \frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}} \exp\left(-\frac{1}{2}(x-\mu)^\intercal \Sigma^{-1}(x-\mu)\right)$$

MLE:

$$\begin{aligned}\hat{\mu}_i &= \text{average value of } x_i \text{ in dataset} \\ \hat{\Sigma}_{i,j} &= \text{average value of } (x_i - \hat{\mu}_i)(x_j - \hat{\mu}_j) \text{ in dataset}\end{aligned}$$